On the Recovery of Cameras from Fundamental Matrices

Rakshith Madhavan and Federica Arrigoni
DEIB – Politecnico di Milano, Italy
rakshith.madhavan@polimi.it, federica.arrigoni@polimi.it

Abstract

The viewing graph is a compact tool to encode the geometry of multiple views: nodes represent uncalibrated cameras and edges represent fundamental matrices (when available). Most research focuses on theoretical analyses, exploring for which viewing graphs it is possible (in principle) to retrieve cameras from fundamental matrices, in the sense that the problem admits a unique solution for noiseless data. However, the practical task of recovering cameras from noisy fundamental matrices is still open, as available methods are limited to special graphs (such as those covered by triplets). In this paper, we develop the first method that can deal with the recovery of cameras from noisy fundamental matrices in a general viewing graph. Experimental results demonstrate the promise of the proposed approach on a variety of synthetic and real scenarios.

1. Introduction

In this paper we consider the task of computing a number of uncalibrated cameras starting from a set of fundamental matrices between pairs of views (see Fig. 1). Cameras are represented as nodes in the so-called *viewing graph* [16], whereas fundamental matrices correspond to the edges. Typically, such a graph is not complete but it contains only a subset of all possible fundamental matrices, which in turn are corrupted by noise/outliers. Exploiting redundancy in the data is the key to achieve error compensation.

The task of camera recovery from fundamental matrices is related to projective *structure from motion* (SfM) [10, 19, 21–23, 29], which aims at recovering both uncalibrated cameras and 3D scene points starting from image point correspondences. Note that fundamental matrices can be derived in closed-form from image points [13], however, there is loss of information when discarding points and using fundamental matrices only as the starting point for projective SfM. This is akin to what happens in the calibrated case with the global SfM approach [1, 25], where essential matrices only (or, equivalently, relative rotations and translation directions) are used as input to perform camera reconstruction. In this respect, the task of camera recovery from fundamental matrices – addressed in this paper – has significant theoretical relevance, as it aims to answer the following question: *what can be done in terms of camera recovery, when we are given only partial information, represented as fundamental matrices?* This problem is poorly studied and highly challenging, as clarified in the subsequent sections.

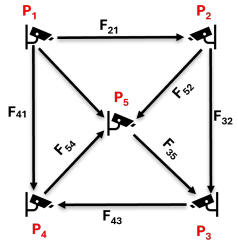


Figure 1. In the camera recovery problem, the task is to recover unknown 3×4 projective camera matrices, $P_1, \dots, P_5$ from the known subset of the fundamental matrices, $\{F_{ij}\}$, between them. The problem has a unique solution, up to a global projective transformation, only if the associated viewing graph is *solvable*.

1.1. Related Work

We first clarify differences between solvability [16], compatibility [7] and camera recovery from fundamental matrices [15], that are tightly related problems. In this paper, we address the latter. All these tasks involve a viewing graph, and they differ on assumptions and objectives on this graph.

- **Solvability**: in this case, we are given a *noiseless* set of fundamental matrices, and the question is *how many* camera configurations are compliant with them. Specifically, it is assumed that a solution (i.e., a set of compatible cameras) exists, and the question is whether such solution is unique or not (up to a single projective transformation). Viewing graphs are typically analyzed using random configurations of cameras, in order to focus on the graph structure itself and understand if the available edges are generically enough to constrain a single solution: in the case of uniqueness (under generic camera configurations), the graph is called *solvable*.

- **Compatibility:** in this case, the question is about the *existence* of a solution, which is typically possible only on *noiseless* data. Hence, the objective is to provide theoretical conditions that characterize a set of compatible cameras in terms of properties of the fundamental matrices.
- **Camera Recovery:** in this scenario, we are given a *noisy* set of fundamental matrices (on a solvable viewing graph), and the question is *how to compute* an approximate solution that minimizes a suitable objective. Typically, an exact solution does not exist with noise.

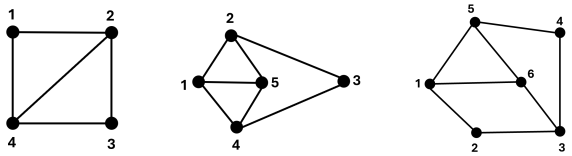
Differences between these problems are summarized in Tab. 1 and more details on existing work are provided in the sequel. Observe that checking solvability permits to identify ill-posed graphs a priori, and it should be done as a preprocessing step *before* camera recovery from fundamental matrices. Compatibility results may inspire practical methods for camera recovery (as done e.g. by [15]), where the idea is to recover cameras while refining fundamental matrices by enforcing compatibility conditions. In this paper, instead, we are not inspired by compatibility theory, but we propose a new iterative scheme for camera recovery.

Table 1. Differences between solvability, compatibility and camera recovery from fundamental matrices.

Solvability	Compatibility	Camera Recovery
noiseless	noiseless	noise
#solutions	existence of a solution	computing a solution

Viewing graph solvability. Existing literature comprises practical necessary conditions [4, 16, 31] or sufficient conditions [26, 30], as well as theoretical characterizations based on polynomial equations [3, 31]. Unfortunately, at the moment, there are no practical methods to check for solvability, in line with the fact that solving polynomial equations is notoriously hard. To overcome this drawback, recent works [2, 5] focused on *relaxations* to the notion of solvability: they check if, for a given graph, the problem of camera recovery from (noiseless) fundamental matrices admits a *finite number of solutions* (instead of a single one), either globally (finite solvability [5]) or locally (infinitesimal solvability [2]). Thanks to the relaxation, the problem is no longer polynomial but it becomes linear (i.e., checking the rank of a proper matrix). Although not being able to distinguish between a single solution and, e.g., two distinct solutions, these methods [2, 5] can be viewed as practical approaches to identify ill-posed graphs with infinitely many solutions, that are the most frequent unsolvable cases in practice. Furthermore, efficient algorithms for partitioning an unsolvable graph into components are available [2, 5].

Compatibility and camera recovery. The first attempt at recovering cameras from a set of noisy fundamental matrices, that we are aware of, was by Sinha et al. [28] for the task of calibrating a network of cameras from silhouettes. From a triplet (3-clique) of fundamental matrices, they simultaneously obtain the three projective cameras and new refined (*consistent*) fundamental matrices between them, using a linear system. They also provide induction steps to extend the approach to larger viewing graphs: this allows recovery of nodes (cameras) belonging to "virtual triplets", arising when adding a node of degree two to the sub-graph containing previously estimated cameras (see Fig. 2b for an example). More recently, Colombo et al. [9] provided a closed-form solution for recovering cameras from a triplet of fundamental matrices, and a sequential extension for more cameras in triplets. Sengupta et al. [27] introduced new (necessary) algebraic rank constraints on a collection of fundamental matrices to be *consistent* (or *compatible*), and optimize the input set of fundamental matrices based on these conditions. They, however, move to the *calibrated* setting for camera recovery so it cannot be considered as an uncalibrated method for recovering cameras from a set of fundamental matrices. This procedure was later extended in GPSFM [15] to introduce more necessary as well as sufficient algebraic constraints for a collection of consistent fundamental matrices. After optimization based on these constraints, Kasten et al. [15] recover projective cameras in triplets, and finally bring them to a common projective frame by concatenating relative projectivities along a spanning tree. GPSFM requires that the nodes of the viewing graph (cameras) are *covered* by triplets, similarly to [9] (see Fig. 2a for an example). Recently, the authors of [18] proposed a framework for *Projectivity Synchronization* to replace the last step of GPSFM (namely the sequential propagation along a spanning tree), exploiting available redundancies. Finally, Bratelund et al. [7] provide a new analysis on *compatibility* of fundamental matrices, extending [15, 27]. However, the results from [7] remain theoretical without developing a practical method for camera recovery.



(a) (b) (c)

Figure 2. Some known solvable viewing graph configurations. All the cameras in (a) can be recovered by [9, 15, 18, 28], since all nodes are covered by triplets. Only [28] can recover all the nodes in (b), where there are two real triplets and one virtual triplet, whereas [9, 15, 18] can only recover a subset of the nodes, namely $\{1,2,4,5\}$. The most general topology of a solvable graph is shown in (c), which can not be managed by previous methods: our approach represents a first step to address this challenge.

1.2. Challenges and Contribution

Overall, the task addressed in this paper – recovering uncalibrated cameras using fundamental matrices only – is poorly studied and counts very few approaches in the literature [9, 15, 18, 28], as reviewed in Sec. 1.1. Most importantly, these methods are limited in the type of viewing graphs they can process: [9, 15, 18] require graphs covered by triplets; [28] considers graphs with a specific topology made of actual triplets and virtual triplets. Examples of graph structures that can be handled by the different methods are shown in Fig. 2. All these cases correspond to solvable graphs [30]. It is important to note that *none of the available methods can manage the most general topology of a solvable viewing graph*: instead, it is known that solvable graphs (for which in principle it is possible to uniquely recover cameras) also include cases without real/virtual triplets [16] (see Fig. 2c).

In fact, recovering cameras from fundamental matrices *on a general viewing graph* is still an open problem. Part of the motivation behind this limited development is that practical methods to check for (approximate) solvability and extract components are extremely recent [2, 5] – note that this is a necessary pre-processing step (see Sec. 1.1). Such works have paved the way to the possibility to build a complete pipeline for camera recovery from fundamental matrices in a general viewing graph. This paper places itself within this context and develops the first approach for such a problem. Note that this task is very challenging: an efficient solution is not available *even in the noiseless case*, where camera recovery can be reduced to a solvability problem described by polynomial equations [4]. Moreover, practical methods that relax solvability [2, 4] can not be used for camera recovery: they analyze an auxiliary linear system that gives information on the *number of solutions* of the original polynomial system, but no information is given on the actual solutions (that are not explicitly retrieved).

Our contributions in addressing these challenges are:

- We propose a novel method for camera recovery from fundamental matrices with an *iterative framework* based on *alternating optimization*, which permits to tackle this challenging optimization task by solving (in alternation) easier problems over subsets of variables/cameras.
- We provide experimental results on a variety of synthetic and real-world data, significantly improving upon the state-of-the-art [9, 15, 18, 28].
- Our approach is the *most general* in the literature, with the ability to manage general graph topologies, provided initialization of a subset of cameras, whereas previous methods were limited to graphs covered by real/virtual triplets.

2. Proposed Formulation

Our task is the following: given a solvable subset out of all possible fundamental matrices between pairs of cameras, the objective is to estimate the n unknown 3×4 camera matrices $P_1, P_2, \dots, P_n$ consistent with them. We model this as the problem of inferring the unknown vertices $\mathcal{V}$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, given the relative measures between them as edges $\mathcal{E}$. This graph is also known as the *viewing graph*, whose vertices are cameras and the edges are the fundamental matrices. We consider an uncalibrated scenario and assume that $\mathcal{G}$ is a *solvable* graph, to ensure that the camera recovery problem is well posed.

To this end, we exploit a popular result from [13] on the consistency relation between two cameras, P_i and P_j, and their fundamental matrix, F_{ij}. Such a result says that F_{ij} is the fundamental matrix of the pair if and only if the matrix $P_i^T F_{ij} P_j$ is skew-symmetric, which is equivalent to:

$$P_i^\mathsf{T} F_{ij} P_j + P_j^\mathsf{T} F_{ij}^\mathsf{T} P_i = 0 \tag{1}$$

where the right side here denotes the 4×4 matrix whose elements are all zero. Note that the above equation involves a single edge $(i, j) \in \mathcal{E}$ of the viewing graph, where F_{ij} is known, while P_i and P_j are unknown. In practice we have multiple equations of the form (1) when considering the whole graph, giving rise to a large polynomial system in unknowns $P_1, \dots, P_n$. Recall that solving Eq. (1) globally is notoriously hard and there are no efficient solutions even in the absence of noise, as explained in Sec. 1.2.

To address this challenge, the key idea is the following: while it is hard to solve Eq. (1) with respect to $P_1, \dots, P_n$ simultaneously, this becomes tractable if we assume that only one camera is unknown but all the others are known; indeed, when considering a single camera as unknown, then Eq. (1) becomes a *linear* system, comprising one set of equations for each neighbor of the unknown camera. To this end, we first rewrite Eq. (1) to its vectorized form as:

$$\big((P_j^\mathsf{T} F_{ij}^\mathsf{T}) \otimes I_4\big)\mathrm{vec}(P_i^\mathsf{T}) + \big(I_4 \otimes (P_j^\mathsf{T} F_{ij}^\mathsf{T})\big)\mathrm{vec}(P_i) = \mathbf{0} \tag{2}$$

where I_4 denotes the 4×4 identity matrix and $\otimes$ refers to the Kronecker product [17], which is a useful tool to isolate one unknown when appearing within a product of matrices[1]. The above equation is equivalent to:

$$\Big(\big((P_j^\mathsf{T} F_{ij}^\mathsf{T}) \otimes I_4\big) K_{3,4} + \big(I_4 \otimes (P_j^\mathsf{T} F_{ij}^\mathsf{T})\big)\Big)\mathrm{vec}(P_i) = \mathbf{0} \tag{3}$$

where $K_{3,4}$ denotes the commutation matrix[2]. Note that, provided that all cameras except P_i are known, then Eq. (3) represents a homogeneous linear system in the canonical form $A_j \mathbf{p_i} = \mathbf{0}$ if the following notation is introduced:

$$\begin{gathered} A_j = \big((P_j^\mathsf{T} F_{ij}^\mathsf{T}) \otimes I_4\big) K_{3,4} + \big(I_4 \otimes (P_j^\mathsf{T} F_{ij}^\mathsf{T})\big) \in \mathbb{R}^{16 \times 12} \\ \mathbf{p_i} = \mathrm{vec}(P_i) \in \mathbb{R}^{12}. \end{gathered} \tag{4}$$

[1] For any matrices A, B, Y of proper dimensions, the Kronecker product satisfies: $\mathrm{vec}(AYB) = (B^\top \otimes A)\mathrm{vec}(Y)$.

[2] The commutation matrix [20], denoted by $K_{t,s}$, is the $ts \times ts$ matrix such that: $\mathrm{vec}(A) = K_{t,s}\mathrm{vec}(A^\mathsf{T})$. The commutation matrix is a permutation, hence it is orthogonal: $K_{t,s} K_{t,s}^\mathsf{T} = I_{ts}$.

This suggests an iterative scheme where, after properly initializing all the cameras, each camera is computed in turn by exploiting linear constraints on its neighbors. This procedure will be formalized in Sec. 3. An alternative (global) approach will be discussed in the supplementary material (Sec. C), which, however, performs poorly.

Remark 1. Observe that, although $A_j\mathbf{p_i} = \mathbf{0}$ defines a system of 16 equations with 12 unknowns (for a single neighbor), only 10 equations are linearly independent: indeed, the starting point of our derivations –Eq. (1) – is tantamount to saying that a 4×4 matrix is skew-symmetric, which defines 10 distinct equations (not 16). In other terms, one neighbor is not enough to have a unique solution, but we need at least two neighbors: this property is automatically satisfied for *solvable* viewing graphs. Indeed, it was proved in [16] that, in a solvable graph, all the vertices have degree at least two and no two adjacent vertices have degree two.

The input fundamental matrices are typically corrupted by noise/outliers, hence a proper cost function must be chosen. We define the global inconsistency as:

$$\Psi(\mathbf{p}_1, \dots, \mathbf{p}_n) = \sum_{(i,j)\in\mathcal{E}} d(A_j\mathbf{p_i}, \mathbf{0}) \tag{5}$$

where d is a suitable distance that evaluates how close the vector $A_j\mathbf{p_i}$ is to the zero vector. Concrete choices for d will be detailed in Sec. 3.1 and 3.2, as well as additional constraints to fix the scale ambiguity for each camera.

Remark 2. Observe that $A_j\mathbf{p_i} = A_i\mathbf{p_j}$, since the left side in Eq. (1) does not change if we exchange i and j. Hence we can equivalently write $d(A_j\mathbf{p_i}, \mathbf{0})$ or $d(A_i\mathbf{p_j}, \mathbf{0})$ in the loss.

To summarize, we aim to minimize the following global objective for the vectorized cameras $\mathbf{p}_1, \dots, \mathbf{p}_n$:

$$\min_{\mathbf{p_1},\dots,\mathbf{p}_n} \Psi(\mathbf{p}_1, \dots, \mathbf{p}_n). \tag{6}$$

3. Proposed Method

In order to address (6), we rely on the framework of *Alternating Optimization* [6] (also known as *Block Relaxation* [11]), which is a recognized tool for turning challenging optimization problems into approachable ones. The procedure is to alternate minimizations restricted over subsets of variables (entries of an individual camera $\mathbf{p_i}$), while keeping the others fixed. In other terms, each node is updated in turn based on its neighbors. An alternating optimization framework was also used in other problems in Computer Vision (such as [12, 18]), showing good performance.

More precisely, the proposed iterative scheme on the viewing graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as follows:

1. Initialize nodes $\mathbf{p_i}^{(0)}$ for $i = 1 \dots n$;
2. For each iteration $k = 1, 2, \dots$ do the following steps:
   (a) For a given node i, estimate $\mathbf{p_i}^{(k)}$ as the minimizer of $\Psi(\mathbf{p_i}^{(k)}) = \Psi(\bar{\mathbf{p_1}}^{(k-1)}, .., \mathbf{p_i}^{(k)}, ..., \bar{\mathbf{p_n}}^{(k-1)})$, where the current camera is treated as the only optimization variable while all the others are kept fixed;
   (b) Repeat Step (a) for all the nodes; for example, for node $(i+1)$, $\mathbf{p_{i+1}}^{(k)}$ is the minimizer of $\Psi(\bar{\mathbf{p_1}}^{(k-1)}, .., \bar{\mathbf{p_i}}^{(k)}, \mathbf{p_{i+1}}^{(k)}, .., \bar{\mathbf{p_n}}^{(k-1)})$;
3. Repeat Step 2 until convergence or a maximum number of iterations is achieved.

For this scheme to work, we need to specify how Step 2 is addressed, namely how to minimize Ψ (with respect to the selected camera), given a set of neighbors and the fundamental matrices between them. Note that, if only camera i is unknown whereas all the others are fixed, then only edges involving node i contribute to the varying cost. Those edges are given by $\{(i,j) \text{ s.t. } j \in \mathcal{N}(i)\}$ where $\mathcal{N}(i)$ denotes the neighbors of node i, which correspond to the adjacent edges. In other words, in Step 2 we have to solve:

$$\min_{\mathbf{p_i}} \sum_{j\in\mathcal{N}(i)} d(A_j\mathbf{p_i}, \mathbf{0}) \tag{7}$$

where A_j, $\mathbf{p_i}$ are defined in Eq. (4) and d is the same distance function as in (5). As will be shown by our experiments in Sec. 4, the simplest idea (i.e., solving a least-squares problem as detailed in Sec. 3.1) yields suboptimal performance, demanding for a more sophisticated technique, which is described in Sec. 3.2. This results in two variants of our approach, described next. See also Sec. 4.1 for implementation details. Some insights about convergence are given in the Supplementary Material (Sec. A).

3.1. The least-squares nullspace problem

In the first proposed method, we solve the homogeneous system of equations from (3) in the least-squares sense. In other terms, we minimize the following objective:

$$\min_{\mathbf{p_i}} \sum_{j\in\mathcal{N}(i)} ||A_j\mathbf{p_i}||_2^2 \quad \text{s.t.} ||\mathbf{p_i}||_2^2 = 1. \tag{8}$$

Note that the optimal $\mathbf{p_i}$ which minimizes this objective, can be retrieved in closed-form. If k denotes the number of neighbors of node i, such solution is found by stacking the 16×12 matrices, A_j, into a $16k \times 12$ matrix A and obtaining the vector $\mathbf{p_i}$ from the SVD decomposition of A. This approach can be made robust to outliers via Iteratively Reweighted Least-Squares (IRLS) [14], as explained in Sec. 4.1. An alternative for robustness could be to employ the L1 norm, however, experimentally we observed no improvement and higher execution time.

3.2. Angular distance to null spaces

As an alternative approach, we make use of the fact that each A_j has a multi-dimensional null space: this derives

from the fact that $A_j\mathbf{p_i} = \mathbf{0}$ does not have a single solution for a single neighbor (see Remark 1). In the noiseless case, these multi-dimensional subspaces (coming from different neighbors) intersect at a unique $\mathbf{p_i}$ (for $k \geq 2$). In the presence of noise, we find the $\mathbf{p_i}$ as the vector that minimizes the angle between such vector and its projection onto the nullspaces of A_j for $j \in \mathcal{N}(i)$. This problem becomes:

$$\min_{\mathbf{p_i}} \sum_{j \in \mathcal{N}(i)} \arccos\left(\frac{\mathbf{p_i}^\mathsf{T} \mathrm{Pr}_{N(A_j)}(\mathbf{p_i})}{||\mathrm{Pr}_{N(A_j)}(\mathbf{p_i})||}\right) \quad \text{s.t.} ||\mathbf{p_i}||_2^2 = 1 \tag{9}$$

where $\mathrm{Pr}_{N(A_j)}(\mathbf{p_i})$ is the *projection*[3] of $\mathbf{p_i}$ onto $N(A_j)$, the nullspace of A_j. The Lagrangian for this problem is:

$$L(\mathbf{p_i}, \lambda) = \sum_{j \in \mathcal{N}(i)} \arccos\left(\frac{\mathbf{p_i}^\mathsf{T} N_{A_j} N_{A_j}^\mathsf{T} \mathbf{p_i}}{||N_{A_j} N_{A_j}^\mathsf{T} \mathbf{p_i}||}\right) + \lambda(1 - \mathbf{p_i}^\mathsf{T} \mathbf{p_i}). \tag{10}$$

Taking the partial derivatives, we get:

$$\frac{\partial L}{\partial \lambda}(\mathbf{p_i}, \lambda) = 1 - \mathbf{p_i}^\mathsf{T} \mathbf{p_i}, \qquad \frac{\partial L}{\partial \mathbf{p_i}}(\mathbf{p_i}, \lambda) = -2\lambda \mathbf{p_i} + \\ - \sum_{j \in \mathcal{N}(i)} \frac{2||B_j \mathbf{p_i}||_2^2 (B_j \mathbf{p_i}) - (\mathbf{p_i}^\mathsf{T} B_j \mathbf{p_i})(B_j^\mathsf{T} B_j \mathbf{p_i})}{(||B_j \mathbf{p_i}||)^4 \sqrt{||B_j \mathbf{p_i}||_2^2 - (\mathbf{p_i}^\mathsf{T} B_j \mathbf{p_i})^2}} \tag{11}$$

where $B_j = N_{A_j} N_{A_j}^\mathsf{T}$. The extrema occur when the partial derivatives are zero, so by setting them to 0 we get:

$$\mathbf{p_i} = \alpha \sum_{j \in \mathcal{N}(i)} \frac{2||B_j \mathbf{p_i}||_2^2 (B_j \mathbf{p_i}) - (\mathbf{p_i}^\mathsf{T} B_j \mathbf{p_i})(B_j^\mathsf{T} B_j \mathbf{p_i})}{(||B_j \mathbf{p_i}||)^4 \sqrt{||B_j \mathbf{p_i}||_2^2 - (\mathbf{p_i}^\mathsf{T} B_j \mathbf{p_i})^2}} \tag{12}$$

where α is a normalizing constant so that $||\mathbf{p_i}||_2^2 = 1$. The sought variable $\mathbf{p_i}$ appears on both sides of the equation, so it is defined as a *fixed point*, which can be estimated iteratively from a good initialization for $\mathbf{p_i}$ (see Sec. 4.1).

4. Experiments

We run all experiments on a Lenovo Legion 5 laptop, with an AMD Ryzen 5 4600H processor and a 16GB RAM. Our iterative framework is implemented in Julia[4]. The two variants of our approach are denoted by LS-SVD (Sec. 3.1), and ANGLE-IT (Sec. 3.2). We compare our approach with previous methods addressing camera recovery from fundamental matrices, discussed in Sec. 1.1: GPSFM [15], GPSFM-SYNCH [18], SINHA [28], and COLOMBO [9]. We use the code supplied by the authors for GPSFM (MATLAB) and GPSFM-SYNCH (Julia+MATLAB), whereas for the methods SINHA and COLOMBO we use our implementation (in Julia), following descriptions in their respective works [9, 28]. Interfacing between MATLAB and Julia is done with a MATLAB system call, and MATLAB functions are called in Julia through the MATLAB.jl library[5].

Observe that the set of cameras recovered from fundamental matrices (with any method) is defined up to a global *projective* transformation. Therefore, in order to compare a solution (denoted by $\hat{P}_1, \ldots, \hat{P}_n$) with the ground-truth cameras (denoted by $P_1, \ldots, P_n$), we have to find a 4×4 homography C such that $\hat{P}_i C \simeq P_i$ for all $i = 1, \ldots, n$. This can be solved linearly [15]. Then, as done in [18], the error for node i is measured as the angular distance between the vectorized estimated and ground truth cameras, brought to a common projective frame: $e_i = \min(\phi, \pi - \phi)$, with

$$\phi = \arccos\left(\mathrm{vec}(\hat{P}_i C), \mathrm{vec}(P_i)\right). \tag{13}$$

4.1. Implementation Details

Regarding Step 1 of our iterative scheme detailed in Sec. 3 (i.e., initialization), we give reasonable starting values[6] to the cameras $P_1, P_2 \ldots P_n$ using the output of GPSFM [15]. Recall that such method requires a graph covered by triplets, which makes it possible that some cameras are not estimated (if they do not belong to any triplet in the viewing graph): we initialize such cameras to the *Canonical Camera* (i.e., the matrix $[I_3 \ \mathbf{0}]$ where $\mathbf{0}$ is the 3×1 vector of zeros). Regarding Step 2, for ANGLE-IT, we initialize $\mathbf{p_i}$ with the vectorized estimate of camera P_i from the previous iteration. More detailed analysis on initialization is given in the Supplementary Material (Sec. B).

Concerning the order in which cameras are updated, we proceed as follows: if some weights associated to the edges in the viewing graph are provided, then we compute the weight of each node by multiplying the weights of its incident edges, and we update the nodes in decreasing order of their weights, based on the idea that higher weights are expected to provide more stable estimates; if we are given an unweighted graph, then nodes are updated in the decreasing order of their centrality measure.

To increase robustness to outliers[7] we embed our approach into an IRLS-like framework, as typically done in geometric estimation problems [8], thereby introducing edge weights w_{ij} in our formulation. The idea is as follows:

1. Set $w_{ij} = 1$ corresponding to each input F_{ij} or compute w_{ij} using steps 3 & 4 with the initial cameras $\{P^0\}$;
2. Compute $\{P_1, \ldots, P_n\} = \text{F2C}(\{F_{ij}\}, \{w_{ij}\})$, where F2C is a function that recovers cameras from fundamen-

[3] The projection is given by $\mathrm{Pr}_{N(A_j)}(\mathbf{x}) = N_{A_j} N_{A_j}^\mathsf{T} \mathbf{x}$, where N_{A_j} is a matrix representing an orthogonal basis of the nullspace of A_j.

[4] The code is available at: https://github.com/rakshith95/cameras_from_F.jl

[5] https://github.com/JuliaInterop/MATLAB.jl

[6] We also tested other possibilities, such as identity and random 3×4 matrices, which performed worse.

[7] Note that LS-SVD (based on squared errors) is not robust to the presence of outliers, but ANGLE-IT is (which minimizes an unsquared error). The IRLS-like framework, hence, has the effect of *introducing* robustness to the former, and *improving* robustness in the case of ANGLE-IT.

tal matrices, implementing one of the two approaches from Sec. 3 (namely LS-SVD , or ANGLE-IT);

3. Compute residuals $r_{ij} = error\big(F_{ij}, f(P_i, P_j)\big)$, where *error*(. , .) is computed as the angle in $\mathbb{R}^9$ between the vectorized matrices, similar to Eq. (13), and $f(P_i, P_j)$ is a function which returns the fundamental matrix given cameras P_i and P_j (see [13]);
4. Update weights as $w_{ij} = w_{ji} = c(r_{ij})$, where c is a robust loss. We used Huber loss in our experiments, i.e. $c(r_{ij}) = \frac{1}{\max(1, |\frac{r_{ij}}{h \cdot c_{\text{huber}} \cdot s}|)}$, where $h = 1, c_{\text{huber}} = 1.345$ and s is computed as the mean absolute deviation *(mad)* of the residuals.
5. Repeat Steps 2 to 4 until convergence or a maximum number of iterations.

4.2. Synthetic Data

In this section we discuss experiments on simulated data.

Data Generation. Given a value n for the number of nodes in the graph, we set up the synthetic environment by generating random ground-truth uncalibrated cameras $P_i \in \mathbb{R}^{3\times4}$ for $i = 1, \dots n$. Concerning the edge set $\mathcal{E}$ in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we remove from the complete graph a random fraction of edges according to a "holes density" $0 \leq \rho < 1$, while ensuring connectivity and finite solvability [5] ($\rho = 0$ produces a complete graph). For each edge $(i, j) \in \mathcal{E}$, we compute its measure F_{ij} from the nodes P_i and P_j with a closed-form expression [13]; such fundamental matrix is then vectorized and normalized to unit norm (thereby representing a 9-dimensional point in the unit sphere) and perturbed by noise. Specifically, noise is introduced following the same protocol used in [18], where $\text{vec}(F_{ij})$ is perturbed by some angle θ_{ij} from a normal distribution with 0 mean and σ standard deviation. Outliers are added by replacing F_{ij} with a random 3×3 matrix R_{ij} of rank 2, with the fraction of outlying edges determined by an outlier density γ ($\gamma = 0$ means zero outliers). These fundamental matrices are given as input to all the analyzed methods. The IRLS scheme is used only in experiments with outliers (i.e., $\gamma > 0$). For each configuration, the test was repeated 100 times and median results were reported. We perform two sets of experiments: in the first case, all nodes in the input graph are covered by triplets and we analyze behaviour of increasing noise, holes, outliers and cameras; in the second case, we simulate *general* graphs, which may or may not contain a real or "virtual" triplet cover.

Noise. For the first set of experiments, we generate graphs ensuring that they are covered by adjacent triplets (those graphs are automatically solvable [30] so we don't explicitly check for solvability). We first analyze the behavior of all competing methods in the presence of increasing noise by varying the standard deviation σ, while keeping the other parameters fixed ($n = 25$, $\rho = 0.4$, $\gamma = 0.0$). Results are reported in Fig. 3 (left), showing the mean of the error over cameras – computed as in (13) – for increasing σ. We observe that our methods report significantly lower errors in comparison with the others, with ANGLE-IT narrowly outperforming LS-SVD. This confirms the effectiveness of our iterative scheme for computing cameras when provided a good initialization. The worst performance is achieved by COLOMBO, due to its sequential nature that tends to accumulate errors. GPSFM, GPSFM-SYNCH and SINHA are more accurate than COLOMBO, as they refine the input fundamental matrices in addition to performing camera recovery.

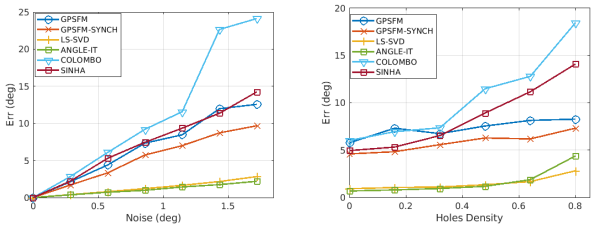


Figure 3. Error [degrees] in camera recovery for several methods. Left: increasing noise σ, with other parameters fixed ($n = 25$, $\rho = 0.4$, $\gamma = 0$). Right: increasing holes density ρ, with other parameters fixed ($n = 25$, $\sigma = 0.015$ rad, $\gamma = 0$). The viewing graph is covered by triplets in these experiments.

Missing Data. Next, we analyze the behavior of the competing methods with respect to varying holes density, ρ, with other parameters fixed: $n = 25$, $\sigma = 0.015$ radians, and $\gamma = 0$. Results are given in Fig. 3 (right), showing that the errors of all methods slightly increase with increasing amount of holes. This is an expected behavior since redundancy helps to achieve error compensation. As before, both variants of our approach outperform the competitors.

Outliers. We also test the effect of varying outlier density, γ, with $n = 25$, $\sigma = 0$, and $\rho = 0.4$, with and without the added robustness from the IRLS-like scheme. We can see in Fig. 4 (left) that GPSFM, GPSFM-SYNCH as well as ANGLE-IT are robust up to 40% outlier density. Indeed, GPSFM performs an optimization on the input fundamental matrices to enforce compatibility constraints before camera recovery, with the effect of removing noise and outliers. GPSFM-SYNCH inherits the same refined fundamental matrices as GPSFM, and it differs in the camera recovery step. Among the two variants of our approach, only ANGLE-IT is robust: indeed, LS-SVD minimizes sum of squares objectives, known to be sensitive to outliers, while ANGLE-IT minimizes a sum of angles instead (unsquared error). SINHA and COLOMBO perform poorly in the presence of outliers, which was also seen in the previous experiments. The effectiveness of our robustness scheme is evident in Fig. 4 (left): the IRLS process significantly imparts robustness to LS-SVD and improves ANGLE-IT.

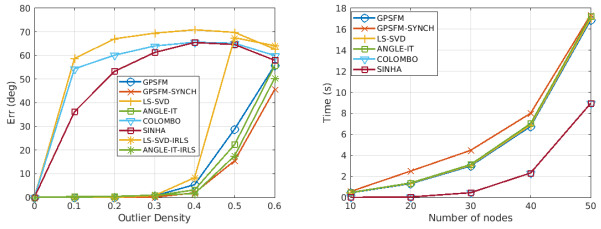


Figure 4. Left: error [degrees] in camera recovery for several methods, for increasing outlier density γ, with other parameters fixed ($n = 25$, $\sigma = 0$, $\rho = 0.4$). Right: execution times [seconds] for increasing number of cameras n, with other parameters fixed ($\sigma = 0.015$ rad, $\rho = 0.4$, $\gamma = 0.0$). The viewing graph is covered by triplets in these experiments.

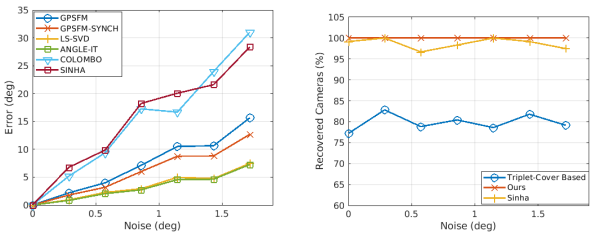


Figure 5. In this experiment, *general* viewing graphs are simulated for increasing noise σ with other parameters fixed ($n = 25$, $\rho = 0.75$, $\gamma = 0$). Left: error [degrees] in camera recovery for several methods (note that errors are averaged only over cameras that have been recovered by each method). Right: the average (mean) percentage of cameras recovered by each type of method: Ours (LS-SVD, ANGLE-IT); Triplet-Cover Based (GPSFM, GPSFM-SYNCH, COLOMBO); SINHA.

Execution Times. We also analyze the efficiency of the methods by reporting the execution times in Fig. 4 (Right). Here the number of cameras n varies, whereas other parameters are fixed ($\sigma = 0.015$ radians, $\rho = 0.4$, $\gamma = 0.0$). Both variants of our approach are comparable to GPSFM: considering that cameras produced by GPSFM are used as initialization (see Sec. 4.1), this shows that our iterative minimization brings negligible additional complexity. The fastest methods are SINHA and COLOMBO, which, however, do not represent valid solutions to the camera recovery problem in terms of accuracy, as shown by previous experiments.

General Graphs. For the second type of experiments, we simulate general graphs randomly, only ensuring finite solvability [5]. We choose to generate relatively sparse graphs, with holes density $\gamma = 0.75$, to increase the likelihood of having nodes not covered by triplets. The other parameters are fixed as $n = 25$ and $\gamma = 0$. We analyze the effect of increasing noise σ in Fig. 5 (left), similarly to the experiment with triplet-based graphs from Fig. 3 (left). However, an important difference is that we do **not** report the mean of the error comparing *all* the estimated nodes $\{\hat{P}_i\}$ with the ground truth nodes, $\{P_i\}$, but we consider *only those cameras which could be recovered by a specific method*. Recall that our framework (LS-SVD, ANGLE-IT) is the only one that can recover all cameras as it processes a general viewing graph by design, whereas previous methods are limited to graphs covered by virtual triplets (SINHA) or real triplets (GPSFM, GPSFM-SYNCH, COLOMBO), as explained in Sec. 1.1. The latter, therefore, can fail to estimate a fraction of cameras. For instance, suppose there are n ground truth nodes out of which m nodes are not estimated: in this case, the error is computed as mean(error($\{\hat{P}_i\}$, $\{P_i\}$)) for $i \in \{1, 2, \ldots, n\} \setminus U$, where U denotes the indices of cameras that are not computed, with $|U| = m$. The percentage of the cameras recovered by all methods is given in Fig. 5 (right). As expected, our method is the only one able to recover 100% of the cameras in all scenarios. Triplet-based methods (GPSFM, GPSFM-SYNCH, COLOMBO), instead, perform poorly in terms of amount of recovered cameras, as they miss about 20% of the cameras on average. SINHA is significantly better than triplet-based techniques, as it also considers virtual triplets in the formulation, but it is still inferior to our approach. Concerning accuracy, given in Fig. 5 (Left), our approach outperforms the competitors.

4.3. Real Data

We consider real datasets used for projective structure from motion, specifically, we test on the same datasets[8] used by the authors of GPSFM [15]: this collection comprises a total of 25 image sequences taken from [24, 32]. For each sequence, the authors of [15] have made available the viewing graphs and the fundamental matrices[9], which are given as input to all the competing methods. Since ground-truth cameras are not available for these data, we consider a different evaluation metric than the synthetic experiments. Specifically, we compute the mean reprojection error (in pixels), achieved after retrieving scene points using the recovered cameras, before Bundle Adjustment.

Results are given in Tab. 2, which also reports the execution times of the competing approaches[10]. We consider three methods in this analysis: GPSFM [15], GPSFM-SYNCH [18] and ANGLE-IT embedded in the IRLS-like scheme – this represents the variant of our approach that gave the best results in the synthetic experiments. We do not report results from SINHA [28] and COLOMBO [9] due to large reprojection error; this is in line with the synthetic experiments with outliers, which cannot be satisfactorily handled by these methods. In this dataset all the viewing graphs

[8]The datasets can be downloaded from https://www.maths.lth.se/matematiklth/personal/calle/dataset/dataset.html

[9]The fundamental matrices can be downloaded from https://github.com/amnonge/GPSFM-code

[10]The execution times for our method includes the time for initialization, which is why it is always higher than GPSFM.

Table 2. Results on real-world projective structure from motion data [24, 32]. For each sequence, the number of nodes/cameras in the viewing graph is reported in addition to its holes density. The reprojection error [pixels] and execution time [seconds], before bundle adjustment, are reported for the competing methods. "Ours" stands for ANGLE-IT-IRLS and GPSFM-SYNCH is abbreviated as SYNCH.

Dataset	Nodes	Holes	Error [px]			Time [s]		
			Ours	SYNCH [18]	GPSFM[15]	Ours	SYNCH [18]	GPSFM[15]
Dino 319	36	0.63	**4.38**	4.89	5.21	2.67	2.43	2.50
Dino 4983	36	0.63	**1.51**	1.59	1.59	2.59	2.55	2.49
Corridor	11	0	**0.49**	0.67	0.71	0.67	0.65	0.64
House	10	0	**1.38**	1.40	1.68	0.59	0.57	0.56
Gustav Vasa	18	0.28	**1.83**	1.99	1.99	1.29	1.29	1.22
Folke Filbyter	40	0.68	**1.78**	1.83	1.96	3.13	2.77	2.81
Park Gate	34	0	**12.81**	13.70	18.45	2.94	2.84	2.73
Nijo	19	0	**17.38**	25.83	29.85	1.47	1.40	1.36
Drinking Fountain	14	0	**1.29**	1.39	1.54	1.00	0.92	0.94
Golden Statue	18	0	**0.82**	0.90	0.95	1.27	1.18	1.21
Jonas Ahls	40	0.59	**28.84**	33.06	36.83	3.08	2.94	2.91
De Guerre	35	0	1.82	1.33	**1.31**	3.04	2.56	2.60
Dome	85	0	4.39	6.40	**4.17**	9.57	8.70	7.80
Alcatraz Courtyard	133	0.08	**13.37**	15.00	36.73	20.66	14.17	14.23
Alcatraz Water Tower	172	0	36.54	19.76	**19.10**	35.86	28.37	29.63
Cherub	65	0.36	**11.92**	14.98	16.87	6.98	5.61	5.41
Pumpkin	195	0.35	**4.58**	9.27	9.57	51.38	40.80	40.87
Sphinx	70	0.67	7.17	**6.16**	8.46	7.04	6.74	6.68
Toronto University	77	0.67	64.10	18.28	**13.85**	8.05	7.37	7.26
Sri Thendayuthapani	98	0	**13.30**	15.12	14.55	12.43	10.29	11.05
Porta San Donato	141	0	**31.81**	41.16	32.41	23.42	18.18	19.70
Buddah Tooth	162	0.27	**16.03**	19.06	25.49	27.70	22.35	21.94
Tsar Nikolai I	98	0.48	10.17	15.26	**9.58**	12.12	11.16	10.25
Smolny Cathedral	131	0	**87.53**	182.80	122.82	23.52	14.88	14.47
Skansen Kronan	131	0.12	**9.02**	10.02	13.71	22.62	15.45	15.39

are actually covered by triplets, therefore all the methods are able to recover all the cameras. Table 2 shows that our approach produces the lowest reprojection error in most cases: more precisely, it outperforms GPSFM in 20 out of 25 sequences, and it is better than GPSFM-SYNCH in 21 out of 25 cases. This highlights the effectiveness of our iterative scheme for camera recovery from fundamental matrices, confirming the outcome of our previous experiments. Concerning execution times, we note that our method, as before, requires little additional time in comparison to GPSFM.

We conclude this section by reporting further analysis on the *Sphinx* sequence. We pruned the input viewing graph by removing some edges, setting a threshold on the number of inlier correspondences. This represents standard practice in structure from motion, where making the graph sparser typically promotes outlier removal. This process produces a new viewing graph which, however, contains one node not appearing in any triplet. In this scenario, both GPSFM and GPSFM-SYNCH estimate 76 out of 77 cameras, failing to estimate the camera not covered by triplets, with mean reprojection errors of 8.16 and 7.10, respectively. Our method, instead, is able to recover *all* the cameras, thanks to the fact that it manages *general* graphs by construction. The error of our method is 4.71 – when considering all cameras – and 4.19 – when considering the subset of 69 cameras covered by triplets, improving over the competitors.

5. Conclusion

In this paper we considered the task of recovering uncalibrated cameras from fundamental matrices. This problem is highly challenging, as there is no reference solution even in the absence of noise, and existing methods are limited to special graphs made by triplets. To fill in this gap, we introduced the first approach that can process a general (solvable) viewing graph, thereby recovering 100% of the cameras even in the case where some cameras do not appear in any triplet, overcoming a significant limitation of previous work. Our method is based on an iterative procedure that solves for a single camera at a time, exploiting the algebraic relationship between a fundamental matrix and the associated camera pair. Experimental results demonstrate superior accuracy than the state of the art in a variety of synthetic scenarios and projective structure from motion sequences. The main limitation of the proposed scheme is that it requires a reasonable initialization for a subset of the cameras. Notwithstanding this, we believe that this work represents a significant advancement towards understanding the hard task of camera recovery from partial information (i.e., no image points but only fundamental matrices), and we hope to inspire more sophisticated methods in the future.

Acknowledgements. This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

References

[1] Federica Arrigoni. A taxonomy of structure from motion methods. *arXiv*, (2505.15814), 2025. 1

[2] Federica Arrigoni, Andrea Fusiello, and Tomas Pajdla. A direct approach to viewing graph solvability. In *Proceedings of the European Conference on Computer Vision*, 2024. 2, 3

[3] Federica Arrigoni, Andrea Fusiello, Elisa Ricci, and Tomas Pajdla. Viewing graph solvability via cycle consistency. In *Proceedings of the International Conference on Computer Vision*, pages 5540 – 5549, 2021. 2

[4] Federica Arrigoni, Andrea Fusiello, Romeo Rizzi, Elisa Ricci, and Tomas Pajdla. Revisiting viewing graph solvability: an effective approach based on cycle consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2022. 2, 3

[5] Federica Arrigoni, Tomas Pajdla, and Andrea Fusiello. Viewing graph solvability in practice. In *Proceedings of the International Conference on Computer Vision*, pages 8147–8155, 2023. 2, 3, 6, 7

[6] D. P. Bertsekas. *Nonlinear programming*. Belmont: Athena Scientific, second edition, 1999. 4

[7] Martin Bratelund and Felix Rydell. Compatibility of fundamental matrices for complete viewing graphs. In *Proceedings of the International Conference on Computer Vision*, pages 3305 – 3313, 2023. 1, 2

[8] Avishek Chatterjee and Venu Madhav Govindu. Robust relative rotation averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5

[9] Carlo Colombo and Marco Fanfani. A closed form solution for viewing graph construction in uncalibrated vision. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2551–2558, 2021. 2, 3, 5, 7, 1

[10] Yuchao Dai, Hongdong Li, and Mingyi He. Projective multiview structure and motion from element-wise factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2238–2251, 2013. 1

[11] Jan de Leeuw. Block-relaxation algorithms in statistics. In Hans-Hermann Bock, Wolfgang Lenski, and Michael M. Richter, editors, *Information Systems and Data Analysis*, pages 308–324, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. 4, 1

[12] R. Hartley, K. Aftab, and J. Trumpf. L1 rotation averaging using the Weiszfeld algorithm. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3041–3048, 2011. 4, 1

[13] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 1, 3, 6

[14] P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977. 4

[15] Y. Kasten, A. Geifman, M. Galun, and R. Basri. GPSfM: Global projective SFM using algebraic constraints on multiview fundamental matrices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3259–3267, 2019. 1, 2, 3, 5, 7, 8

[16] Noam Levi and Michael Werman. The viewing graph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 518 – 522, 2003. 1, 2, 3, 4

[17] Shuangzhe Liu and Gotz Trenkler. Hadamard, Khatri-Rao, Kronecker and other matrix products. *International Journal of Information and Systems Sciences*, 4(1):160 – 177, 2008. 3

[18] Rakshith Madhavan, Andrea Fusiello, and Federica Arrigoni. Synchronization of projective transformations. In *European Conference on Computer Vision (ECCV)*, 2024. 2, 3, 4, 5, 6, 7, 8, 1

[19] Ludovic Magerand and Alessio Del Bue. Revisiting projective structure from motion: A robust and efficient incremental solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):430–443, 2020. 1

[20] J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, revised edition, 1999. 3

[21] Dror Moran, Hodaya Koslowsky, Yoni Kasten, Haggai Maron, Meirav Galun, and Ronen Basri. Deep permutation equivariant structure from motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5976–5986, 2021. 1

[22] Behrooz Nasihatkon, Richard Hartley, and Jochen Trumpf. A generalized projective reconstruction theorem and depth constraints for projective factorization. *International Journal of Computer Vision*, 115:87 – 115, 2015.

[23] J. Oliensis and R. I. Hartley. Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2217–2233, 2007. 1

[24] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *Proceedings of the 17th Scandinavian conference on Image analysis (SCIA'11)*, pages 524–535. Springer-Verlag, 2011. 7, 8

[25] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305 – 364, 2017. 1

[26] Alessandro Rudi, Matia Pizzoli, and Fiora Pirri. Linear solvability in the viewing graph. In *Proceedings of the Asian Conference on Computer Vision*, pages 369–381, 2011. 2

[27] S. Sengupta, T. Amir, M. Galun, T. Goldstein, D. W. Jacobs, A. Singer, and R. Basri. A new rank constraint on multiview fundamental matrices, and its application to camera location recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2413–2421, 2017. 2

[28] S.N. Sinha, M. Pollefeys, and L. McMillan. Camera network calibration from dynamic silhouettes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages I–I, 2004. 2, 3, 5, 7, 1

[29] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proceedings of the European Conference on Computer Vision*, pages 709–720, 1996. 1

[30] M. Trager, M. Hebert, and J. Ponce. The joint image handbook. In *Proceedings of the International Conference on Computer Vision*, pages 909–917, 2015. 2, 3, 6

[31] Matthew Trager, Brian Osserman, and Jean Ponce. On the solvability of viewing graphs. In *Proceedings of the European Conference on Computer Vision*, pages 335–350, 2018. 2

[32] Visual Geometry Group - University of Oxford. Multiview datasets. https://www.robots.ox.ac.uk/~vgg/data/. 7, 8